Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

ALGEBRA MODULARE E CIFRE DI CONTROLLO PER IL RILEVAMENTO DI ERRORI

Tesi di Laurea in Topologia Algebrica

Relatore:
Chiar.mo Prof.
MASSIMO FERRI

Presentata da:
ELEONORA GRIDELLI

III Sessione
Anno Accademico 2016-2017

"Me and Sam in the car, talking 'bout America
Heading to the wishing well, we've reached our last resort
I turned to him, said: "Man help me out
I fear I'm on an island in an ocean full of change
Can't bring myself to dive in to an ocean full of change.
Am I losing touch
Am I losing touch now?"
[...]
Hey pretty smiling people, we're alright together
We're alright together
Hey pretty shining people, we're alright together
We're alright together, hey"
(Pretty Shining People, George Ezra)

Alla mia famiglia,
Ai miei amici.

Indice

Introduzione

Tutte le informazioni sono generalmente codificate in sequenze di bit. Esse possono riguardare prodotti, documenti, persone e molto altro e, per assicurare la loro affidabilità e completezza, è necessario corredarle con delle cifre di controllo, o check digit. Questi ultimi devono almeno permettere il riconoscimento degli errori semplici, frequenti e non malevoli, cioé non intenzionali (non parleremo di attacchi informatici), che avvengono durante la registrazione, la lettura o la trasmissione dei dati via computer.

Le possibili tipologie di codice per mantenere l'integrità dei dati sono due: quelli a rilevazione di errore e quelli a correzione di errore. I primi rintracciano la presenza di errori nei dati ricevuti, dovuti a rumore (segnale casuale che perturba il canale di trasmissione influendo sulla qualità delle informazioni), distrazioni umane o altre cause esterne, mentre i secondi sono, inoltre, in grado di ricostruire i dati originali, ricalcolandoli a partire da quelli ricevuti.

Per determinare i check digit si considerano le stringhe di codice da proteggere e le si elaborano in modi diversi: si possono fare semplici calcoli sulle sequenze dei dati, applicare nozioni di algebra modulare o implementare algoritmi che variano per costo e complessità della matematica utilizzata. La modalità con cui manipolare i dati è scelta anche in base ai risultati che ci si propone di raggiungere, ossia quali errori vogliamo trovare e in che percentuale.

In questa tesi illustreremo come i gruppi possano essere utilizzati per realizzare metodi in grado di rintracciare gli errori e, di conseguenza, garantire l'affidabilità delle informazioni ricevute. Tali metodi saranno differenziati in base agli strumenti, alle conoscenze a priori e agli obiettivi prefissati.

Così, nel primo capitolo verrà in primo luogo mostrato il tipo di codice più semplice, ovvero quello per il sistema binario, lo stesso cui si rifanno i metodi attualmente implementati nei computer. Esso è in grado di riconoscere quando una stringa viene perturbata con un errore e, in questo caso, di chiedere la sua ritrasmissione. Nell'eventualità in cui i bit modificati siano due, non è sempre vero che il sistema sia capace di rintracciarli. Tuttavia questa è una situazione piuttosto rara dal momento che la probabilità p di avere un singolo errore è per definizione minore di 1, per cui la probabilità di averne due sarà piccola, pari a p^2.

Ci addentreremo poi in ambienti leggermente più complessi: andremo a considerare le tipologie di errori che occorrono nel momento in cui si hanno a disposizione più di due caratteri e cercheremo di comprendere l'importanza dello studio della frequenza di tali errori, non tanto per l'individuazione e la correzione, quanto per la prevenzione.

Nel secondo capitolo viene invece riportata tutta la teoria su cui si poggiano i metodi che vedremo nel terzo capitolo ovvero l'algebra modulare e le trasformazioni nel piano. Nello specifico tratteremo dei gruppi ciclici e dei gruppi diedrali.

Infine, il terzo capitolo è dedicato alle applicazioni: riprenderemo gli argomenti trattati in precedenza e li caleremo nel nostro caso specifico. Mostreremo le loro conseguenze concrete e ne sfrutteremo le proprietà per studiare una ad una le tecniche di calcolo della cifra di controllo per l'individuazione degli errori.

Presenteremo infine degli esempi, partendo dai metodi più semplici e deboli per arrivare a quelli più completi ed efficaci.

Capitolo 1

Codici e check digit

Un check digit è una cifra decimale, o alfanumerica, che viene aggiunta a un numero con lo scopo di individuare tutti i possibili errori che generalmente si compiono nell'inserimento manuale dei dati e nella loro trasmissione.

1.1 Bit di parità

Per prima cosa consideriamo la codifica binaria utilizzata negli apparecchi elettronici come i computer in cui tutte le informazioni sono rappresentate tramite sequenze di bit. Il bit è l'unità elementare di informazione e corrisponde alla presenza oppure all'assenza di un segnale elettrico, due stati rappresentati rispettivamente dai valori **1** e **0**.

Il segnale può tuttavia essere perturbato e generare degli errori, in particolare, nel momento del trasporto dei dati possono avvenire degli errori di trasmissione (ricezione di dati distorti, non conformi ai dati inviati) causati sia da fattori interni al sistema, come la perdita di sincronizzazione tra trasmettitore e ricevitore, sia da fattori esterni, come l'interferenza di un campo magnetico sul canale trasmissivo. Diventa quindi necessario un intervento per controllare la validità e la consistenza dei dati. Esistono diversi metodi, quello più semplice è il controllo di parità. Esso si basa sull'aggiunta di informazione ridondante, il bit di parità appunto: durante la scrittura dei dati (sequenze di bit) si contano gli **1** presenti in ogni byte e, se sono in

numero dispari, si aggiunge un **1**, se sono in numero pari, uno **0** in modo da avere *bit parity*.

Il bit di parità viene quindi calcolato come la somma aritmetica modulo 2 dei bit nella sequenza. Siano $i_1, i_2, \ldots, i_n$ i bit di informazione, il bit di parità p è dato da

$$p = \left(\sum_{k=1}^{n} i_k \right)_{mod\ 2}$$

Supponendo che il bit di controllo non possa essere modificato, il codice è in grado di rilevare solo un numero dispari di errori, infatti un singolo errore produce un numero dispari di **1** quindi una codifica non valida. Questo evidenzia la sua fallibilità: il metodo non riconosce né la modifica di due bit con lo stesso valore né, più in generale, la modifica simultanea di due cifre perché la sequenza trasmessa continuerà a contenere un numero pari di **1**. Il bit di parità è purtroppo adatto solo per la rilevazione degli errori per cui se arriva una sequenza errata, viene scartata e deve essere richiesta la ritrasmissione dell'intera sequenza.

D'altronde è ragionevole assumere che la probabilità che un errore occorra sia molto minore di 1, di conseguenza la probabilità che ne occorrano due sarà notevolmente più piccola perché corrisponde al suo quadrato. Nonostante la sua semplicità e vulnerabilità, quindi, il metodo del bit di parità è uno dei più utilizzati perché ha il vantaggio di essere un codice che usa un solo bit di spazio ed è poco costoso (anche se risulta lento per lunghe sequenze).

I contenuti di questo paragrafo sono tratti da [5] e [6].

1.2 Rilevamento degli errori

Gli errori non sono uniformemente distibuiti, nel senso che alcuni ricorrono più spesso di altri. Quelli più comuni sono elencati nella tabella 1.1 [2].

Per individuare tali errori i bit di informazione originali vengono corredati di un check digit. Ci sono molti metodi per il sistema decimale (esistono anche per sistemi alfanumerici) e ognuno di essi è in grado di identificare tutti gli errori di cifra singola mentre solitamente fallisce con quelli di trasposizione. In caso occorrano altri tipi di errore non è meccanico che il metodo li determini o meno. È noto però

che ogni procedura che individua i *single digit error* è automaticamente in grado di individuare circa il 90% degli errori di formato, un altro errore molto comune dato dall'inserimento o dalla cancellazione di una o più cifre (4711→47811 o 4711→411) [3]. L'uso di due check digit non è comunque consigliato poiché il numero di errori quasi raddoppia quando il numero di bit di controllo aumenta di due. Vediamo qualche esempio.

L'*Universal Product Code*, abbreviato in UPC, è un tipo specifico di codice a barre usato per il tracciamento di articoli commerciali. È costituito da dodici cifre numeriche nell'insieme $\{0, 1, \ldots, 9\}$, le prime sei identificano la nazione e il produttore, le cinque successive il prodotto e l'ultima è la cifra di controllo calcolata sfruttando l'aritmetica modulare. Leggermente più complicato è invece lo schema sviluppato da IBM, usato per lo più da società di carte di credito, biblioteche e banche del sangue, che sfrutta anche le permutazioni su n numeri. Per dettagli su come vengano calcolate le cifre di controllo dei due metodi rimandiamo al Capitolo 3.

Sia lo schema UPC sia quello dell'IBM rintracciano il 100% dei *single digit errors* ma nessuno dei due è capace di individuare la totalità dei *transposition errors*. In particolare, un errore del tipo 3ab48→3ba48 passa inosservato per UPC quando $|a - b| = 5$ e per IBM quando $(a, b) = (0, 9)$ o $(9, 0)$. Segue che UPC rileva gli errori di trasposizione di cifre adiacenti l'88,9% delle volte mentre la percentuale di successo per lo schema di IBM sarà del 97,8% [2].

Error Type	Form	Relative Frequency
single digit error	a→b	79.1%
trasposition of adjacent digits	ab→ba	10.2%
jump transposition	abc→cba	0.8%
twin error	aa→bb	0.5%
phonetic error[1]	0a→1a, a=0,...,9	0.5%
jump twin error	aca→bcb	0.3%

Tabella 1.1: Common pattern errors

[1]per i numeri letti in lingua inglese come 15 e 50

1.3 Correzione e prevenzione degli errori

Abbiamo visto che per evitare che la modifica di un messaggio trasmesso su un canale rumoroso passi inosservata, si usano metodi specifici per individuarla che però si limitano a constatare l'avvenuta o la mancata perturbazione del dato e non tentano in nessun modo di rintracciare l'errore nella sequenza. Invece, nella maggior parte dei casi, piuttosto che richiedere una ritrasmissione completa della sequenza, è conveniente usare un tipo di codice in grado di calcolare il messaggio originale a partire dal messaggio corrotto che viene ricevuto. Questo tipo di codice è detto *a correzione d'errore*. Vediamo nuovamente l'esempio dato dalla bit parity: aggiungendo altri check digit e sfruttando un meccanismo sofisticato è in grado di localizzare il bit corrotto nella sequenza e cambiare il suo valore da **0** a **1** e viceversa.

Nonostante questi sistemi per individuare e/o correggere gli errori, si svela un'altra necessità, quella della prevenzione, da non tralasciare sia per metodi semplici sia per metodi più avanzati. Per prevenzione si intende cercare di ridurre il numero di errori agendo sulle diverse probabilità di comparsa degli errori dei diversi tipi, come abbiamo visto nella tabella 1.1. Quindi, in generale, data la non uniformità di distribuzione degli errori nelle sequenze di bit, se scegliamo un codice in modo che la totale possibilità di errore sia minima, abbiamo già ottenuto un certo livello di prevenzione dell'errore [9].

Infine, un altro fatto poco conosciuto è l'incremento della possibilità di errore aggiungendo check digit alla sequenza di dati. Da una parte sembrerebbe che avere più ridondanza (ovvero più check digit) potrebbe ridurre la percentuale degli errori non individuati, in realtà l'effetto è quasi opposto: diminuisce anche la percentuale di bit corretti. Infatti più è lungo il dato, meno informazioni la rilevazione dell'errore porta [4].

Capitolo 2

Gruppi

La teoria dei gruppi è centrale in matematica e viene usata in diversi ambiti. In questo capitolo ne sarà riportata una minima parte che comprederà le nozioni essenziali e alcune semplici strutture utili al nostro percorso.

Definizione 2.1. Un'*operazione binaria*, o *legge di composizione*, su un insieme G è un'applicazione

$$G \times G \longrightarrow G$$
$$(a, b) \longmapsto a \circ b$$

dove ad (a, b) è assegnato in modo unico $a \circ b$, detto *composizione* di a e b.

Definizione 2.2. Un *gruppo* è una coppia $(G, \circ)$, dove G è un insieme non vuoto e $\circ$ è una legge di composizione, tale che:

- la legge di composizione è associativa ovvero $(a \circ b) \circ c = a \circ (b \circ c)$ per ogni $a, b, c \in G$
- esiste un elemento $e \in G$, chiamato *elemento neutro*, tale che per ogni elemento $a \in G$ si ha $e \circ a = a \circ e = a$
- per ogni elemento $a \in G$, esiste l'*elemento inverso* in G, indicato con a^{-1}, tale che $a \circ a^{-1} = a^{-1} \circ a = e$

Un gruppo G in cui $a \circ b = b \circ a \ \forall a, b \in G$ è detto *abeliano* o *commutativo*.

A volte siamo interessati a gruppi più piccoli all'interno di gruppi più grandi.

Definizione 2.3. Diciamo che H è un *sottogruppo* di G quando $H \subseteq G$ come sottinsieme e H è un gruppo con la legge di composizione di G ristretta a H.

Proposizione 2.0.1. *Un sottinsieme H di G è un sottogruppo se e solo se soddisfa le seguenti condizioni:*

i) se $h_1, h_2 \in H$, allora $h_1 \circ h_2 \in H$

ii) se $h \in H$, allora $h^{-1} \in H$.

Proposizione 2.0.2. *Un sottinsieme H di un gruppo G è suo sottogruppo se e solo se $H \neq \emptyset$, e $g, h \in H \Rightarrow gh^{-1} \in H$.*

Teorema 2.0.3. *Sia G un gruppo e sia a un qualsiasi elemento di G. Allora l'insieme*

$$< a >= \{a^k \mid k \in \mathbb{Z}\}$$

è un sottogruppo di G. In particolare, $< a >$ è il più piccolo sottogruppo di G contenente a.

Definizione 2.4. Sia $a \in G$, chiamiamo $< a >$ il *sottogruppo ciclico* generato da a. Se G contiene un elemento a tale che $G = < a >$, allora G è un *gruppo ciclico* e a è detto *generatore* di G.

Definizione 2.5. Si dice che un gruppo è *finito*, o è di *ordine finito*, quando ha un numero finito di elementi; altrimenti diciamo che il gruppo è *infinito*, o che ha *ordine infinito*. L'*ordine* di un gruppo è il numero di elementi che contiene. Se G è un gruppo che contiene n elementi, scriviamo $|G| = n$, altrimenti $|G| = \infty$.

I contenuti di questo paragrafo sono tratti da [1] e [7].

2.1 Gli interi mod n

Definizione 2.6. Siano r e s due interi e supponiamo $n \in \mathbb{N}$. Diciamo che r è *congruente a s modulo n* se $r - s$ è divisibile per n ovvero $r - s = nk$ per un certo $k \in \mathbb{Z}$. Scriviamo $r \equiv s \pmod{n}$.

La congruenza modulo n è una relazione d'equivalenza in $\mathbb{Z}$ infatti:

1. è riflessiva poiché ogni intero r è equivalente a se stesso, ovvero $r - r = 0$ è divisibile per n

2. è simmetrica poiché se $r \equiv s \pmod{n}$ allora $r - s = -(s - r)$ è divisibile per n, quindi $s - r$ è divisibile per n e possiamo scrivere $s \equiv r \pmod{n}$

3. è transitiva poiché se $r \equiv s \pmod{n}$ e $s \equiv t \pmod{n}$ allora esistono interi k e l tali che $r - s = kn$ e $s - t = ln$. Mostriamo che $r - t$ è divisibile per n: $r - t = r - s + s - t = kn + ln = (k + l)n$ quindi possiamo scrivere $r \equiv t \pmod{n}$.

Gli interi modulo n formano una partizione di $\mathbb{Z}$ in n diverse classi. Chiamiamo questa partizione *insieme delle classi resto mod n* e la indichiamo con $\mathbb{Z}_n$. Possiamo fare dell'aritmetica su $\mathbb{Z}_n$. Dati due interi $a, b \in \mathbb{Z}_n$ definiamo l'addizione mod n come $(a + b) \pmod{n}$; analogamente la moltiplicazione modulo n è definita come $(ab) \pmod{n}$.

$\cdot$	**0**	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Tabella 2.1: Tabella di moltiplicazione di $\mathbb{Z}_5$

Gli interi mod n formano un gruppo con l'addizione mod n.

Proposizione 2.1.1. *Sia $\mathbb{Z}_n$ l'insieme delle classi di equivalenza degli interi mod n e siano $a, b, c \in \mathbb{Z}_n$.*

i) Addizione e moltiplicazione sono commutative:

$$a + b \equiv b + a \quad (mod\ n)$$
$$ab \equiv ba \quad (mod\ n).$$

ii) Addizione e moltiplicazione sono associative:

$$(a+b)+c \equiv a+(b+c) \pmod{n}$$
$$(ab)c \equiv a(bc) \pmod{n}.$$

iii) Ci sono sia l'elemento neutro additivo sia quello moltiplicativo, che indichiamo rispettivamente con 0 *e* 1*:*

$$a+0 \equiv a \pmod{n}$$
$$a \cdot 1 \equiv a \pmod{n}.$$

iv) La moltiplicazione è distributiva rispetto all'addizione:

$$a(b+c) \equiv ab+ac \pmod{n}.$$

v) Per ogni intero a esiste l'inverso additivo $-a$*:*

$$a+(-a) \equiv 0 \pmod{n}.$$

vi) Sia $a \in \mathbb{Z}^*$ *allora mcd*(a,n)*=1 se e solo se esiste un inverso moltiplicativo b per a (mod n), ovvero un intero non nullo tale che*

$$ab \equiv 1 \pmod{n}.$$

I contenuti di questo paragrafo sono tratti da [7].

2.2 Le simmetrie

Le simmetrie delle figure piane vengono solitamente classificate in: simmetria bilaterale, simmetria di rotazione (figura 2.1), simmetria di traslazione e simmetria di glissoriflessione (figura 2.2).

Possono inoltre presentarsi altre combinazioni di simmetrie, per esempio la stella possiede sia la simmetria bilaterale sia la simmetria di rotazione (figura 2.3).

Figura 2.1: Esempi di riflessione e rotazione

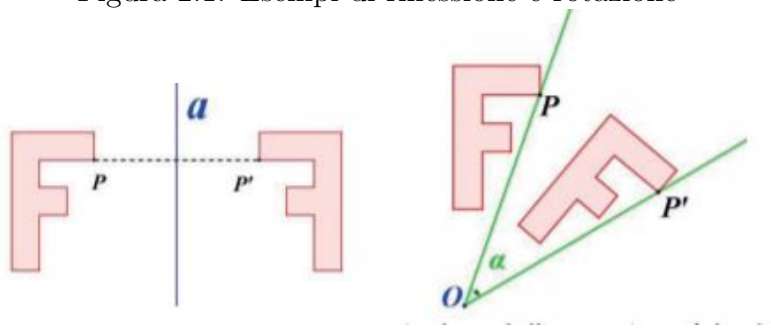


Figura 2.2: Esempi di traslazione e glissoriflessione

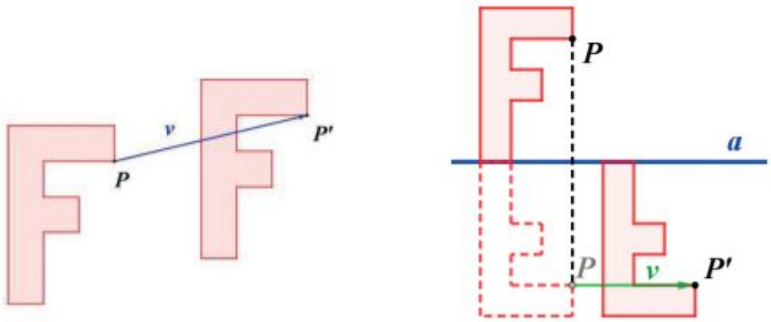
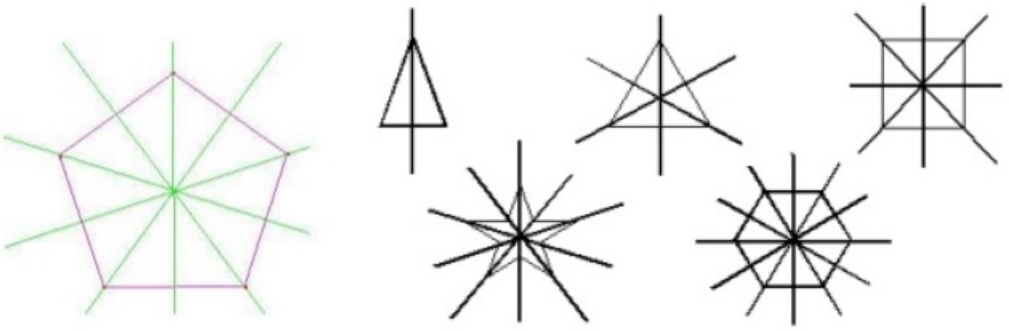


Figura 2.3: Esempi di figure che godono di più di un tipo di simmetria



Definizione 2.7. Si dice che un'applicazione $m : P \longrightarrow P$ del piano P in sé è un *movimento rigido* o un'*isometria*, se conserva le distanze, ovvero se dati comunque due punti $p, q \in P$, la distanza da p a q è uguale alla distanza da $m(p)$ a $m(q)$.

Definizione 2.8. Un movimento rigido m che porta un sottinsieme F del piano in sé è detto *simmetria* di F. Una simmetria di una figura geometrica è una nuova

disposizione della figura che ne preserva la forma.

Mostreremo nel prossimo paragrafo che i movimenti rigidi sono proprio le traslazioni, le rotazioni, le riflessioni e le glissoriflessioni e che essi formano un gruppo M con la composizione di applicazioni. Inoltre l'insieme di tutte le simmetrie di F costituisce sempre un sottogruppo G di M, detto *gruppo delle simmetrie della figura* F. Per esempio, il gruppo delle simmetrie delle figure dotate di simmetria bilaterale è formato da due elementi: l'identità e la simmetria attorno a una retta chiamata asse di simmetria (è quindi un gruppo ciclico di ordine 2).

Passiamo ora a descrivere il gruppo M di tutti i movimenti rigidi del piano. Ci sono movimenti che conservano e movimenti che invertono l'orientazione del piano: in base a questa prima partizione grossolana possiamo definire un omomorfismo di gruppi

$$M \longrightarrow \pm 1$$

che manda i movimenti che conservano l'orientazione in 1 e quelli che la invertono in -1.

Operiamo ora una classificazione più fine:

1. movimenti che conservano l'orientazione:
   - traslazione: movimento parallelo del piano mediante un vettore a: $p \mapsto p + a$
   - rotazione: movimento che ruota il piano di un angolo $\theta \neq 0$ intorno a un punto

2. movimenti che invertono l'orientazione:
   - riflessione intorno a una retta l
   - glissoriflessione: movimento ottenuto componendo una riflessione intorno a una retta l e una traslazione mediante un vettore non nullo a parallelo a l.

Teorema 2.2.1. *L'elenco precedente comprende tutti i casi possibili. Ogni movimento rigido del piano è una traslazione o una rotazione, o una riflessione, o una glissoriflessione, o l'identità.*

Per poter svolgere agevolmente i calcoli nel gruppo M, scegliamo alcuni movimenti particolari come generatori per il gruppo. Identifichiamo il piano con lo spazio

$\mathbb{R}^2$ dei vettori colonna scegliendo un sistema di coordinate, poi prendiamo come generatori le traslazioni, le rotazioni intorno all'origine e la riflessione intorno all'asse x_1:

1. traslazione t_a di vettore a: $t_a(x) = x + a = \begin{bmatrix} x_1 + a_1 \\ x_2 + a_2 \end{bmatrix}$
2. rotazione ρ_θ di un angolo θ intorno all'origine: $\rho_\theta(x) = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
3. riflessione r intorno all'asse x_1: $r(x) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ -x_2 \end{bmatrix}$.

Poiché lasciano fissa l'origine, le rotazioni ρ_θ e la riflessione r sono operatori ortogonali su $\mathbb{R}^2$. Una traslazione non è un operatore lineare (non manda il vettore nullo in sé) a meno che non si tratti della traslazione con vettore nullo. I movimenti elencati non sono tutti gli elementi di M, tuttavia essi generano il gruppo, ossia ogni elemento di M è prodotto di questi elementi. Ogni elemento di $m \in M$ può essere ottenuto da essi per composizione come $m = t_a\rho_\theta$ oppure $m = t_a\rho_\theta r$, per qualche vettore a e angolo θ eventualmente nulli. Tale espressione è unica.

Per poter ridurre un qualsiasi prodotto di generatori a una delle forme appena citate occorrono opportune regole di composizione in modo da effettuare più agevolmente calcoli in M:

$$\begin{aligned} &t_a t_b = t_{a+b}, \quad \rho_\theta\rho_\eta = \rho_{\theta+\eta}, \quad rr = id, \\ &\rho_\theta t_a = t_{a'}\rho_\theta, \quad \text{dove} \quad a' = \rho_\theta(a), \\ &r t_a = t_{a'} r, \quad \text{dove} \quad a' = r(a), \\ &r\rho_\theta = \rho_{-\theta} r. \end{aligned} \tag{2.1}$$

Quindi, per quanto detto precedentemente, vi sono due sottogruppi notevoli di M:

T, gruppo delle traslazioni

$\boldsymbol{O}$, gruppo degli operatori ortogonali

Il gruppo $\boldsymbol{O}$ è costituito dai movimenti che lasciano fissa l'origine, ovvero le rotazioni attorno all'origine e le riflessioni intorno alle rette passanti per l'origine. Esiste

un omomorfismo notevole φ da M a $\boldsymbol{O}$, con nucleo T (e non dipende dalla scelta dell'origine)

$$\begin{aligned} \varphi : M &\longrightarrow \boldsymbol{O} \\ t_a\rho_\theta &\longmapsto \rho_\theta \\ t_a\rho_\theta r &\longmapsto \rho_\theta r \end{aligned}$$

Tramite le formule (2.1) è facile dimostrare che è effettivamente un omomorfismo:

$$(t_a\rho_\theta)(t_b\rho_\eta) = t_a t_{b'}\rho_\theta\rho_\eta = t_{a+b}\rho_{\theta+\eta},$$

da cui:

$$\varphi(t_a\rho_\theta t_b\rho_\eta) = \rho_{\theta+\eta},$$

analogamente per le altre operazioni. Si noti che non è possibile definire in modo analogo un omomorfismo da M a T.

Studiamo ora i possibili gruppi finiti di simmetrie ovvero i sottogruppi finiti G del gruppo M dei movimenti rigidi del piano.

Teorema 2.2.2 (Teorema del punto fisso)**.** *Sia G un sottogruppo finito del gruppo dei movimenti M. Allora esiste un punto p nel piano che è lasciato fisso da ogni elemento di G, ossia esiste un punto p tale che $g(p) = p$ per ogni $g \in G$.*

Dimostrazione. Vediamo la dimostrazione da un punto di vista geometrico. Sia s un punto del piano e sia S l'insieme dei punti che sono immagini di s rispetto ai vari movimenti di G. Pertanto ogni elemento $s' \in S$ sarà della forma $s' = g(s)$, con $g \in G$. Tale insieme è chiamato *orbita* di s rispetto all'azione di G. L'elemento s appartiene all'orbita, poiché id sta in G e $s = id(s)$.

Ogni elemento del gruppo G permuta l'orbita S. In altre parole, se $s' \in S$ e $x \in G$ allora $x(s') \in S$. Infatti, sia $s' = g(s)$, con $g \in G$. Poiché G è un gruppo, $xg \in G$. Pertanto, per definizione, $xg(s) = x(s')$, ciò prova che $x(s') \in S$.

Elenchiamo ora gli elementi di S in un ordine arbitrario, $S = \{s_1, s_2, \ldots, s_n\}$; il punto fisso che stiamo cercando è proprio il *centro di gravità* dell'orbita definito da

$$p = \frac{1}{n}(s_1 + s_2 + \cdots + s_n), \tag{2.2}$$

dove l'espressione al secondo membro è calcolata mediante l'addizione tra vettori, in un arbitrario sistema di coordinate nel piano. Il centro di gravità dovrebbe essere considerato come una *media* dei punti $s_1, s_2 \ldots, s_n$. Per terminare la dimostrazione abbiamo bisogno di un lemma.

Lemma 2.2.3. *Sia $S = \{s_1, s_2, \ldots, s_n\}$ un insieme finito di punti del piano, sia p il suo centro di gravità, definito da (2.2). Sia m un movimento rigido, poniamo $m(s_i) = s_i'$ e $m(p) = p'$. Allora $p' = \frac{1}{n}(s_1' + s_2' + \cdots + s_n')$. In altre parole, movimenti rigidi mandano centri di gravità in centri di gravità.*

Dimostrazione. Trattiamo separatamente i casi $m = t_a$ (traslazione di vettore a), $m = \rho_\theta$ (rotazione di angolo θ) e $m = r$ (riflessione) poiché ogni movimento si ottiene da questi per composizione.

Caso 1: $m = t_a$. Allora $p' = p + a$ e $s_i' = s_i + a$, quindi:

$$p + a = \frac{1}{n}((s_1 + a) + (s_2 + a) + \cdots + (s_n + a)).$$

Caso 2: $m = \rho_\theta$ oppure $m = r$. Allora m è un operatore lineare, pertanto:

$$p' = m\left(\frac{1}{n}(s_1 + s_2 + \cdots + s_n)\right) = \frac{1}{n}(m(s_1) + m(s_2) + \cdots + m(s_n))$$
$$= \frac{1}{n}(s_1' + s_2' + \cdots + s_n').$$

□

Il centro di gravità dell'insieme S è un punto fisso rispetto all'azione di G. Infatti un elemento arbitrario g_i di G permuta l'orbita $\{s_1, s_2, \ldots, s_n\}$ cosicché il lemma 2.2.3 prova che esso manda il centro di gravità in sé. Questo completa la dimostrazione del teorema. □

Sia ora G un sottogruppo finito di M. Il teorema 2.2.2 ci assicura che esiste un punto lasciato fisso da ogni elemento di G. Possiamo scegliere le coordinate in modo che questo punto sia l'origine. Allora G risulterà un sottogruppo di $\boldsymbol{O}$. Pertanto, per descrivere i sottogruppi finiti G di M, basta descrivere i sottogruppi finiti di $\boldsymbol{O}$.

Teorema 2.2.4. *Sia G un sottogruppo finito del gruppo $\boldsymbol{O}$ dei movimenti rigidi che lasciano fissa l'origine. Allora G è uno dei gruppi seguenti:*

i) $G = C_n$, *il gruppo ciclico di ordine n, generato dalla rotazione* ρ_θ, *dove* $\theta = 2\pi/n$,

ii) $G = D_n$, *il gruppo diedrale di ordine 2n, generato da due elementi: la rotazione* ρ_θ, *dove* $\theta = 2\pi/n$, *e una riflessione* r' *intorno a una retta per l'origine.*

Rimandiamo la dimostrazione del teorema a più avanti nel paragrafo per poter fare alcune precisazioni.

Il gruppo D_n dipende dalla retta di riflessione, ma possiamo scegliere le coordinate in modo che essa diventi l'asse x e allora r' diventa la riflessione standard r. Se pensiamo G come sottogruppo finito di M, dobbiamo prima spostare l'origine nel punto fisso per poter applicare il teorema. Il risultato finale è dato dal seguente corollario.

Corollario 2.2.5. *Sia* G *un sottogruppo finito del gruppo dei movimenti* M. *Allora, introdotto un sistema di coordinate opportuno,* G *diventa uno dei gruppi* C_n *o* D_n, *dove il primo è generato da* ρ_θ, *con* $\theta = 2\pi/n$, *e il secondo è generato da* ρ_θ *e* r.

Per $n \geq 3$ il gruppo diedrale D_n è il gruppo delle simmetrie di un poligono regolare di n lati infatti un poligono regolare di n lati ha un gruppo di simmetria che contiene la rotazione di angolo $2\pi/n$ intorno al suo centro e alcune riflessioni (figura 2.4). Per il teorema 2.2.4 tale gruppo è proprio D_n.

Abbiamo un insieme di relazioni anche per i gruppi diedrali (si ricavano direttamente dalle relazioni (2.1) che definiscono M). Denotiamo la rotazione ρ_θ, dove $\theta = 2\pi/n$, con x e la riflessione r con y.

Proposizione 2.2.6. *Il gruppo diedrale* D_n *è generato da due elementi* x, y *che soddisfano le relazioni:*

$$x^n = 1, \quad y^2 = 1, \quad yx = x^{-1}y \,^1 \tag{2.3}$$

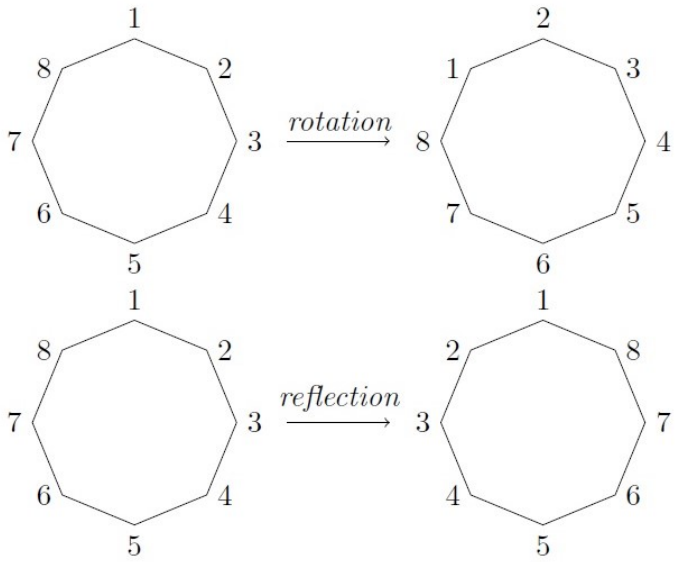
Gli elementi di D_n *sono:*

$$\{1, x, x^2, \ldots, x^{n-1}; y, xy, x^2y, \ldots, x^{n-1}y\} = \{x^i y^j \mid 0 \leq i < n,\ 0 \leq j < 2\}.$$

[1] usando le prime due possiamo riscriverla come $yx = x^{n-1}y$ oppure $xyxy = 1$

Dimostrazione. Gli elementi $x = \rho_\theta$ e $y = r$ generano il gruppo D_n, per definizione. Le relazioni $y^2 = 1$ e $yx = x^{-1}y$ sono incluse nell'elenco delle relazioni (2.1) per M: esse sono $rr = 1$ e $r\rho_\theta = \rho_{-\theta}r$. La relazione $x^n = 1$ segue dal fatto che $\theta = 2\pi/n$, il che prova anche che gli elementi $1, x, \ldots, x^{n-1}$ sono distinti, e quindi, poiché essi sono riflessioni, mentre le potenze di x sono rotazioni, non vi sono ripetizioni nella lista degli elementi. Infine le relazioni possono essere usate per ridurre un prodotto arbitrario di x, y, x^{-1}, y^{-1} nella forma $x^i y^j$ con $0 \leq i < n$, $0 \leq j < 2$. Pertanto la lista contiene tutti gli elementi del gruppo generato da x, y e, poiché tali elementi generano D_n, la lista è completa. □

Figura 2.4: Gruppo delle simmetrie di un poligono regolare con otto lati



Si noti che per $n = 3$ il gruppo diedrale D_3 e il gruppo simmetrico S_3 sono tra loro isomorfi, mentre per $n > 3$ il gruppo diedrale e il gruppo simmetrico certamente non sono isomorfi, poiché D_n ha ordine $2n$, mentre S_n ha ordine $n!$, in particolare D_n è sottogruppo di S_n.

Dimostrazione. (del teorema 2.2.4). Sia G un sottogruppo finito di $\boldsymbol{O}$. Occorre ricordare che gli elementi di $\boldsymbol{O}$ sono le rotazioni ρ_θ e le riflessioni $\rho_\theta r$.

Caso 1: tutti gli elementi di G sono rotazioni, proviamo che G è ciclico. Se $G = \{1\}$, allora $G = C_1$, altrimenti G contiene una rotazione non banale ρ_θ. Sia θ il più piccolo angolo di rotazione positivo tra gli elementi di G, allora G è generato da ρ_θ. Infatti, sia ρ_α un elemnto arbitrario di G, dove l'angolo di rotazione α è rappresentato da un numero reale. Sia $n\theta$ il più grande multiplo intero di θ minore di α, sicché $\alpha = n\theta + \beta$, con $0 \leq \beta < \theta$. Poiché G è un gruppo e ρ_α e ρ_θ sono elementi di G, anche il prodotto $\rho_\beta = \rho_\alpha \rho_{-n\theta}$ è un elemento di G. Ma, per ipotesi, θ è il più piccolo angolo di rotazione positivo in G, quindi $\beta = 0$ e $\alpha = n\theta$. Ciò prova che G è ciclico. Sia $n\theta$ il più piccolo multiplo di θ che sia $\geq 2\pi$, sicché $2\pi \leq n\theta < 2\pi + \theta$. Poiché θ è il più piccolo angolo di rotazione positivo in G, si ha: $n\theta = 2\pi$. Dunque $\theta = 2\pi/n$ per qualche intero n.

Caso 2: G contiene una riflessione; a meno di un cambiamento di coordinate, possiamo assumere che sia la riflessione standard r ad appartenere a G. Indicando con H il sottogruppo delle rotazioni in G, possiamo applicare ciò che è stato dimostrato nel caso 1 al gruppo H, per concludere che esso è un gruppo ciclico, ovvero $H = C_n$. Allora i $2n$ prodotti ρ_θ^i, $\rho_\theta^i r$, $0 \leq i \leq n-1$, appartengono a G, quindi G contiene il gruppo diedrale D_n. Dobbiamo mostrare che $G = D_n$. Ora se un elemento g di G è una rotazione, allora $g \in H$ per definizione di H, dunque g è un elemento di D_n. Se g è una riflessione, allora g può essere scritto nella forma $\rho_\alpha r$ per qualche rotazione ρ_α. Poiché r è un elemento di G, tale risulta il prodotto $\rho_\alpha r r = \rho_\alpha$. Pertanto ρ_α è una potenza di ρ_θ, e anche g appartiene a D_n, quindi $G = D_n$. □

I contenuti di questo paragrafo sono tratti da [1].

Capitolo 3

Applicazioni dei gruppi ai check digit

I check digit vengono solitamente calcolati attraverso una formula o un algoritmo e, in base a certe condizioni, possiamo metterli a confronto. È noto che non è possibile stabilire a priori e in assoluto i requisiti per un buon codice: essi dipendono non solo dal tipo di equipaggiamento materiale a disposizione, ma anche dalla conoscenza degli errori abituali degli esseri umani coinvolti. Sicuramente, i metodi che vengono proposti devono essere in grado di risolvere (o perlomeno risolvere nella maggioranza dei casi) alcuni tipi di errori; noi chiediamo che abbiano successo con quelli che abbiamo precedentemente citato, ovvero contro i *single digit errors* con massima priorità e subito dopo contro i *transposition errors*, contro i *twin errors* e i *jump transposition errors* con media priorità e contro i *jump twin errors* e i *phonetic errors* con priorità minima.

Consideriamo i gruppi di ordine 10: sappiamo dalla teoria dei gruppi che ne esistono di due tipi, quello ciclico C_{10} e quello diedrale D_5.

C_{10} è il gruppo delle rotazioni di un decagono regolare ed è abeliano mentre D_5 è il gruppo delle trasformazioni di un pentagono regolare e non è commutativo. L'idea di sfruttare i gruppi sta nel fatto che grazie alle proprietà della legge di composizione associativa vengono notevolmente semplificate le condizioni per il rilevamento degli errori.

Vediamo ora una carrellata di metodi per assegnare un check digit a un numero dato e studiarne il comportamento con gli errori più comuni.

3.1 Codici con addizione modulo 10

I primi metodi che vedremo sono le generalizzazioni del controllo di parità. Esse si inseriscono nel sistema dei numeri decimali e sono quindi controlli in aritmetica modulo 10. Se chiamiamo *parola* una sequenza di bit, questo codice consisterà di tutte le parole che soddisfano $a_1 + a_2 + \cdots + a_n = p \pmod{10}$, dove p è il check digit. Ovviamente questo metodo individua tutti gli errori di carattere singolo ma, anche se coglie l'88.9% dei *twin errors* (poiché $2a \equiv 2b \pmod{10}$ se $a \equiv b + 5 \pmod{10}$), non individua nessun errore di trasposizione. Gli errori di fonetica vengono invece rintracciati il 100% delle volte.

Se un numero n è composto dalle cifre $d_k, d_{k-1}, \ldots, d_1$ in base r, ovvero $n = \sum_{i=1}^{k} d_i r^{i+1}$ e scriviamo la cifra di controllo come

$$p = -\sum_{i=1}^{k} d_i \pmod{r}$$

allora p sarà in grado di rintracciare tutti gli errori singoli. Il numero "protetto" sarà quindi rappresentato da $d_k, d_{k-1}, \ldots, d_1, p$ e, per analizzarne la correttezza, basterà verificare che la somma delle cifre sia congrua a zero modulo n. Il motivo per il quale un errore di trasposizione non verrà mai individuato è la commutatività della somma modulo r, quindi è bene assegnare dei pesi w_i alle posizioni delle cifre prima di sommarle. Se la sequenza dei pesi è $(w_0, w_1, \ldots)$ allora scegliamo p tale che

$$w_k d_k + w_{k-1} d_{k-1} + \cdots + w_1 d_1 + w_0 p \equiv 0 \pmod{10}.$$

Esempio 3.1. Prendiamo come pesi $(w_0, w_1, \ldots) = (1, 3, 7, 9, 1, 3, \ldots)$ e come numero da proteggere $n = 90148323$ avremo

9	0	1	4	8	3	2	3	p	digits
1	9	7	3	1	9	7	3	1	weights
$9+0+7+12+8+27+14+9+p$									$\equiv 6+p \pmod{10}$.

Per ottenere 0 (mod 10) dobbiamo porre $p = 4$, cosicché il numero codificato sarà 901483234.

Notiamo però che la trasposizione $\ldots 83 \ldots \to \ldots 38 \ldots$ nel numero dell'esempio non viene rintracciata. Si potrebbe pensare di cambiare la sequenza dei pesi ma è chiaro che 1, 3, 7, 9 sia l'unica sequenza possibile per individuare gli errori singoli; infatti pesi consecutivi w_i, w_{i+1} non possono essere uguali tra loro e nessuno di essi può essere pari perché se w_i fosse pari avremmo $w_i \cdot d_i = w_i \cdot d_i'$ ogni volta che $(d_i - d_i') \equiv 5 \pmod{10}$, di conseguenza, non ci sarebbe distinzione tra i caratteri d_i e d_i'. Analogamente, w_i non può essere pari a 5. Possiamo generalizzare dicendo che per individuare gli errori di cifra singola i w_i devono essere coprimi con 10.

Invece, gli errori di trasposizione non vengono mai trovati quando $(d_i - d_{i-1}) \equiv 5 \pmod{10}$. In questo caso, infatti, dato che i w_i sono dispari, $w_i - w_{i-1}$ risulta essere pari, quindi

$$(w_i - w_{i-1})(d_i - d_{i-1}) = (w_i - w_{i-1}) \cdot 5 \equiv 0 \pmod{10}$$
$$\ldots w_i d_i + w_{i-1} d_{i-1} \ldots = \ldots w_i d_{i-1} + w_{i-1} d_i \ldots$$

Se r è primo, è evidente che scegliendo dei numeri $a_i \in \{1, \ldots, r-1\}$ con $a_i \neq a_{i+1}$ e definendo $\tau_i(x) := a_i x \pmod{r}$, il metodo rintraccia, come vedremo, tutti gli errori di trasposizione, mentre se r è pari, in generale il metodo fallisce, studieremo poi un caso specifico.

Nonostante i difetti, questo metodo detto *weighted parity check method*, fornisce l'idea chiave di trasformare un digit x in posizione i in un qualcosa del tipo $w_i x$.

Lo step successivo sarà quindi quello di scegliere delle trasformazioni per ogni posizione nella sequenza, ovvero una permutazione di $D = \{0, 1, \ldots, 9\}$.

Imponiamo permutazioni τ_i dei caratteri $\{0, 1, \ldots, r-1\}$ su ogni posizione: calcoleremo p in modo che $\tau_k(d_k) + \cdots + \tau_1(d_1) + \tau_0(p) \equiv 0 \pmod{r}$. Solitamente si sceglie $\tau_0 = id$ così da poter risolvere l'equazione per p:

$$p = -\sum_{i=1}^{k} \tau_i(d_i) \pmod{r}.$$

Se scegliamo per ogni $x \neq y$ e per ogni posizione i, $\tau_i(x) \neq \tau_i(y)$ siamo sicuri di individuare tutti gli errori di single digit mentre la capacità di trovare quelli di trasposizione sarà determinata dalla scelta della sequenza $(\tau_1, \tau_2, \dots)$.

Scegliere una particolare permutazione rispetto a un'altra non risolverebbe comunque totalmente il problema in quanto τ_i, τ_{i-1} dovrebbero soddisfare, per esempio, la relazione

$$\tau_i(d_i) + \tau_{i-1}(d_{i-1}) \neq \tau_i(d_{i-1}) + \tau_{i-1}(d_i) \pmod{10}$$

per ogni coppia $d_i \neq d_{i-1}$, con $d_i, d_{i-1} \in \{0, \dots, 9\}$, cosa che non è possibile. Vediamo quello che succede in realtà.

Proposizione 3.1.1. *Ogni metodo con permutazioni modulo 10 non è in grado di rintracciare tutti gli errori di trasposizione* $\dots ab \dots \to \dots ba \dots$

Dimostrazione. Se dobbiamo rintracciare una trasposizione $\dots ab \dots \to \dots ba \dots$ nelle posizioni $i, i+1$, abbiamo bisogno che

$$\tau_{i+1}(a) + \tau_i(b) \neq \tau_{i+1}(b) + \tau_i(a) \pmod{10},$$

e quindi

$$\tau_{i+1}(a) - \tau_i(a) \neq \tau_{i+1}(b) - \tau_i(b) \pmod{10}.$$

Denotiamo la trasformazione $x \mapsto \tau_{i+1}(x) - \tau_i(x)$ con τ, avremo che $\tau(a) \neq \tau(b) \pmod{10}$. Questo deve essere valido $\forall a \neq b$ quindi τ è una permutazione di $\{0, \dots, 9\}$. Ma il lemma seguente mostra che una τ definita come sopra non può mai essere una permutazione, avremo quindi la conclusione della dimostrazione. □

Lemma 3.1.2. *Se α e β sono permutazioni sull'insieme $\{0, \dots, 9\}$, allora la loro differenza $\tau = \alpha - \beta \pmod{10}$ non sarà mai una permutazione.*

Dimostrazione. Dato che α, β sono iniettive, avremo

$$D = \{0, \dots, 9\} = \{\alpha(x) | x \in D\} = \{\beta(x) | x \in D\}.$$

Se τ è una permutazione vale la stessa cosa, quindi

$$D = \{\tau(x) | x \in D\} = \{\alpha(x) - \beta(x) | x \in D\}.$$

Ma in questo modo si giunge a una contraddizione, ovvero prendendo la somma modulo 10 di tutti gli elementi di questi insiemi avremo:

$$5 = \sum_{x \in D} x = \sum_{x \in D} \tau(x) = \sum_{x \in D} (\alpha(x) - \beta(x)) = \sum_{x \in D} \alpha(x) - \sum_{x \in D} \beta(x) = 0.$$

□

La dimostrazione è in realtà valida per ogni sistema numerico con base $r = 2k$.

Definizione 3.1. Un *metodo di check digit su base* r è dato da un insieme D, con $|D| = r$, e da una famiglia $F := (f_1, f_2, \dots, f_n, \dots)$ di operazioni $f_n : D^n \longrightarrow D$ tali che i seguenti assiomi siano soddisfatti per ogni $n \in \mathbb{N}$:

$$f_n(x_n, \dots, x_i, \dots, x_1) = f_n(x_n, \dots, x_i', \dots, x_1) \Rightarrow x_i = x_i' \tag{3.1}$$

$$f_n(x_n, \dots, x_i, x_{i-1}, \dots, x_1) = f_n(x_n, \dots, x_{i-1}, x_i, \dots, x_1) \Rightarrow x_i = x_{i-1} \tag{3.2}$$

$$f_n(x_n, \dots, x_2, f_n(x_n, \dots, x_1)) = x_1 \Rightarrow f_n(x_n, \dots, x_1) = x_1. \tag{3.3}$$

Chiaramente, $f_k(d_k, \dots, d_1)$ è da interpretare come la cifra di controllo per il numero che è rappresentato dalle cifre $d_k, \dots, d_1$ in base r. Senza perdita di generalità possiamo inoltre assumere che $D = \{0, 1, \dots, r-1\}$. Naturalmente l'assioma (3.1) garantisce il rilevamento degli errori di cifra singola, mentre (3.2) e (3.3) garantiscono per gli errori di trasposizione. In particolare, (3.3) serve a individuare le trasposizioni della cifra di controllo con la cifra vicina alla sua sinistra. La definizione appena data specifica un'algebra $A = (D, F)$.

Dato che gli assiomi sono universali, i risultati standard dell'algebra ci dicono che essi vengono conservati dal prodotto interno. Questo risultato è esplicitato dalla seguente proposizione.

Proposizione 3.1.3. *Se esistono metodi di check digit con base* r *e con base* s*, allora esiste un metodo di check digit con base* rs*.*

Per la dimostrazione basta scrivere ogni cifra in base rs con una coppia (d_1, d_2), dove d_1 è in base r e d_2 è in base s. Quindi si calcolano i check digit p_1 e p_2 di entrambe le componenti, separatamente, nelle loro rispettive basi e si riconverte la

coppia (p_1, p_2) nel corrispondente numero in base rs. Infine si verifica che le proprietà (3.1), (3.2) e (3.3) si conservano.

Dato che ci sono metodi di check digit per la base q, con q primo dispari, la proposizione 3.1.3 fornisce metodi per ogni base con un numero primo dispari.

Avvertiamo che esistono modi anche per ottenere dei metodi di check digit in base r, con r potenza di un primo (anche nel caso $r = 2^m$ con $m > 1$).

I contenuti di questo paragrafo sono tratti da [3] e [4].

3.2 Un diverso tipo di addizione

Consideriamo ora i metodi di check digit per i numeri in base $r = 2s$, con s primo dispari. Arriveremo a dare un metodo in cui s è un qualsiasi numero dispari più grande o pari a tre.

Come precedentemente mostrato un controllo pesato è impossibile da realizzare, dobbiamo quindi intervenire in altro modo sostituendo l'addizione con un tipo diverso di operazione. Chiamiamo questa operazione $*$ e andiamo a calcolare la cifra di controllo p come prima:

$$\tau_k(d_k) * \tau_{k-1}(d_{k-1}) * \cdots * \tau_1(d_1) * \tau_0(p) = 0. \tag{3.4}$$

Sicuramente le τ_i dovranno essere iniettive e di conseguenza saranno permutazioni dell'insieme $\{0, \ldots, r-1\}$. Ora, supponiamo che $a * b = a * c$ per qualche cifra $a, b, c \in \{0, \ldots, 9\}$ e consideriamo una posizione i. Siano $a' = \tau_i^{-1}(a)$, $b' = \tau_{i-1}^{-1}(b)$, $c' = \tau_{i-1}^{-1}(c)$, avremo

$$\tau_i(a') * \tau_{i-1}(b') = \tau_i(a') * \tau_{i-1}(c')$$

cioè un errore che scambia b' e c' non sarebbe rintracciato. Quindi vogliamo che $b' = c'$, da cui $b = c$. Quindi il primo accorgimento su $*$ è che dovrà valere la legge di cancellazione da entrambe le parti, ovvero $a * x = a * y$ implica $x = y$ e $x * a = y * a$ implica $x = y$ per ogni $a, x, y \in \{0, 1, \ldots, r-1\}$.

Se scriviamo l'operazione $*$ in una tabella 10×10 dove nella casella di riga i-esima e colonna j-esima troviamo il prodotto $i * j$, gli accorgimenti precedenti implicano che: (a) le entrate di ogni riga sono reciprocamente differenti, (b) le entrate di

ogni colonna sono reciprocamente differenti. Una tabella di questo tipo è chiamata *quadrato latino*.

I quadrati latini 10×10 sono un quantitativo enorme, inoltre per essere sicuri di "sommare" più di due numeri senza specificare l'ordine dell'operazione ed evitando l'uso di parentesi abbiamo bisogno che $*$ sia una legge associativa ovvero $a*(x*y) = (a*x)*y$. Quindi, dato che agisce sull'insieme finito $\{0, 1, \ldots, r-1\}$, $*$ deve essere un'operazione di gruppo. In questo caso, dovremo definire un elemento neutro 0 tale che $0*x = x*0 = x$ per ogni x e un inverso x^{-1} tale che $x^{-1}*x = x*x^{-1} = 0$ per ogni $x \in \{0, \ldots, 9\}$.

Nel caso in cui $r = 2p$, con p primo dispari, ci sono precisamente due gruppi di r elementi, ossia il gruppo ciclico $\mathbb{Z}_r$ e il gruppo diedrale D_p.

In $\{0, \ldots, 9\}$ il primo è dato dall'addizione modulo 10 e il secondo è il gruppo diedrale di ordine 10 dotato della tabella di "addizione" seguente. Notiamo che l'operazione non è commutativa, per esempio $3*6 \neq 6*3$, ma per i nostri scopi, ovvero il rilevamento degli errori di trasposizione, non sarà sicuramente uno svantaggio.

$*$	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	0	6	7	8	9	5
2	2	3	4	0	1	7	8	9	5	6
3	3	4	0	1	2	8	9	5	6	7
4	4	0	1	2	3	9	5	6	7	8
5	5	9	8	7	6	0	4	3	2	1
6	6	5	9	8	7	1	0	4	3	2
7	7	6	5	9	8	2	1	0	4	3
8	8	7	6	5	9	3	2	1	0	4
9	9	8	7	6	5	4	3	2	1	0

Tabella 3.1: Tabella di "addizione" per il gruppo D_5

Generalizzando, siano s un numero dispari maggiore di 2 e $r = 2s$. Il gruppo diedrale D_s di ordine r può essere rappresentato come l'insieme delle coppie (e, x)

con $e \in \{-1, 1\}$ e $x \in \{0, 1, \ldots, s-1\}$, dove $*$ è definito nel modo seguente

$$(e, x) * (f, y) := (ef, ey + x)$$

e la moltiplicazione e l'addizione interne sono valutate nel gruppo ciclico $\mathbb{Z}_s$ di ordine s. Questa notazione è un'abbreviazione per la rappresentazione matriciale

$$\begin{pmatrix} e & x \\ 0 & 1 \end{pmatrix}$$

degli elementi di D_s dove $*$ è la moltiplicazione tra matrici.

Ora rivolgiamoci alle τ_i, vogliamo definirle in modo che anche gli errori di trasposizione vengano rintracciati ovvero per ogni a, b e per ogni posizione i vogliamo

$$\tau_i(a) * \tau_{i-1}(b) = \tau_i(b) * \tau_{i-1}(a) \Rightarrow a = b.$$

Poniamo $u := \tau_{i-1}(a)$ e $v := \tau_{i-1}(b)$ e $\tau := \tau_i \tau_{i-1}^{-1}$. Quindi l'errore di trasposizione è individuato se troviamo una sequenza τ_i di permutazioni tali che ogni $\tau := \tau_i \tau_{i-1}^{-1}$ soddisfa l'equazione $\tau(u) * v = \tau(v) * u \Rightarrow u = v$. Per $a, b \in \mathbb{Z}_s$ con $a \neq 0$ definiamo un'applicazione $\tau : D_s \longrightarrow D_s$ tale che

$$\tau(e, x) = (e, e(a - x) + b).$$

Allora avremo il seguente lemma.

Lemma 3.2.1. *τ è una permutazione in D e per ogni $u, v \in D_s$ soddisfa l'implicazione $\tau(u) * v = \tau(v) * u \Rightarrow u = v$.*

Dimostrazione. Se $u = (e, x)$ e $v = (f, y)$, allora $\tau(u) = \tau(v)$ implica $e = f$ e $e(a-x)+b = f(a-y)+b$. Quindi $x = y$, allora τ è una permutazione. Supponiamo che $\tau(u) * v = \tau(v) * u$, quindi con la stessa notazione

$$(e, e(a - x) + b) * (f, y) = (f, f(a - y) + b) * (e, x)$$

quindi

$$(ef, ey + e(a - x) + b) = (ef, fx + f(a - y) + b).$$

Da cui

$$e(a+y-x) = f(a+x-y) \quad \text{ovvero} \quad (e-f)a = (e+f)(x-y).$$

Se $e \neq f$, allora il termine a destra si annulla e otteniamo $2a = 0$, che però è impossibile dato che s è dispari e la moltiplicazione è quella del gruppo ciclico $\mathbb{Z}_s$. Perciò $e = f$ che, di nuovo, implica $2(x-y) = 0$ così $x = y$. □

Ora, dato che abbiamo la permutazione cercata, basta porre $\tau_0 = id$ e $\tau_i := \tau^i = \tau \circ \tau_{i-1}$ dato che $\tau_i \circ \tau_{i-1}^{-1} = \tau^i \circ (\tau^{i-1})^{-1} = \tau^{i-(i-1)} = \tau$.

Possiamo quindi descrivere un metodo di check digit per $r = 2s$ usando l'applicazione τ e le sue iterate $\tau, \tau^2, \tau^3, \ldots$.

Teorema 3.2.2. *Per qualsiasi τ presa come nel lemma 3.2.1, la definizione*

$$f_n(x_n, \ldots, x_1) := [\tau^n(x_n) * \tau^{n-1}(x_{n-1}) * \cdots * \tau(x_1)]^{-1}$$

determina un metodo di check digit in base $r = 2s$.

Dimostrazione. Verifichiamo gli assiomi della definizione 3.1, partendo direttamente dal secondo.

$$\begin{aligned}\left[\tau^n(x_n) * \cdots * \tau^{i+1}(x_{i+1}) * \tau^i(x_i) * \tau^{i-1}(x_{i-1}) * \tau^{i-2}(x_{i-2}) * \cdots * \tau(x_1)\right]^{-1} = \\ \left[\tau^n(x_n) * \cdots * \tau^{i+1}(x_{i+1}) * \tau^i(x_{i-1}) * \tau^{i-1}(x_i) * \tau^{i-2}(x_{i-2}) * \cdots * \tau(x_1)\right]^{-1}.\end{aligned}$$

Da cui, svolgendo le inversioni e cancellando i termini uguali, otteniamo

$$\tau^i(x_i) * \tau^{i-1}(x_{i-1}) = \tau^i(x_{i-1}) * \tau^{i-1}(x_i),$$

quindi

$$\tau(\tau^{i-1}(x_i)) * \tau^{i-1}(x_{i-1}) = \tau(\tau^{i-1}(x_{i-1})) * \tau^{i-1}(x_i).$$

Usando il lemma otteniamo $\tau^{i-1}(x_i) = \tau^{i-1}(x_{i-1})$ e siccome τ è iniettiva otteniamo $x_i = x_{i-1}$. Per il terzo assioma dovremo fare qualche calcolo in più. Assumiamo che $f_n(x_n, \ldots, x_2, f_n(x_n, \ldots, x_1)) = x_1$ ovvero

$$\left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau\left(\left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1}\right)\right]^{-1} = x_1$$

Usando le leggi del gruppo deduciamo

$$\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau\left(\left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1}\right) = x_1^{-1}$$

$$\tau\left(\left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1}\right) = \left[\tau^n(x_n) * \cdots * \tau^2(x_2)\right]^{-1} * x_1^{-1}.$$

Moltiplicando con $\tau(x_1)^{-1}$ otteniamo

$$\tau(x_1)^{-1} * \tau\left(\left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1}\right) = \left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1} * x_1^{-1}$$

$$\tau(x_1)^{-1} * \tau\left(\left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1}\right) * x_1 = \left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1}$$

$$\tau\left(\left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1}\right) * x_1 = \tau(x_1) * \left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1}$$

Per il lemma otteniamo

$$\left[\tau^n(x_n) * \cdots * \tau^2(x_2) * \tau(x_1)\right]^{-1} = x_1, \quad \text{ovvero} \quad f_n(x_n, \ldots, x_1) = x_1.$$

□

Combinando il precedente teorema con la proposizione 3.1.3 otteniamo il risultato seguente.

Teorema 3.2.3. *Per ogni numero $r \neq 2$ esiste un metodo di check digit per il sistema numerico in base r.*

In più, se r è un numero composto solitamente esistono più di una decomposizione in prodotti per r che identificano altrettanti metodi di check digit e, dato che il metodo mostrato permette modifiche, vediamo un corollario al teorema.

Corollario 3.2.4. *Sia $r = 2s$ con s dispari, sia $(\tau_1, \tau_2, \dots, \tau_n, \dots)$ una sequenza di permutazioni di $\{0, 1, \dots, r-1\}$ che soddisfano il lemma. Definiamo $\alpha_1 := \tau_1$ e $\alpha_i := \tau_i \circ \alpha_{i-1}$ per $i > 1$, allora*

$$f_n(x_n, x_{n-1}, \dots, x_1) := [\alpha_n(x_n) * \alpha_{n-1}(x_{n-1}) * \alpha_1(x_1)]^{-1}$$

determina un metodo di check digit in base r. Inoltre, se la cifra 0 in base r è assegnata alla coppia $(1, 0)$ nella rappresentazione del gruppo diedrale D_s e se scegliamo $a = -b$ nella definizione dell'applicazione τ, otteniamo $\tau(0) = 0$, per cui avere o non avere zeri iniziali nel numero da codificare non avrà effetto nel calcolo della cifra di controllo.

Teorema 3.2.5. *Sia $*$ l'operazione del gruppo diedrale sulle cifre nell'insieme $D = \{0, 1, \dots, 9\}$ e sia τ la permutazione su D che soddisfa il lemma, allora scegliere p tale che*

$$\tau^n(d_n) * \tau^{n-1}(d_{n-1}) * \cdots * \tau(d_1) * p = 0$$

determina un metodo di check digit che individua tutti gli errori di cifra singola e di trasposizione.

Considerando D_5, il metodo è semplice da usare e implementare su un computer. Rappresentiamo gli elementi $(1, x)$ e $(-1, x)$ con le cifre decimali x e $5 + x$ rispettivamente. Abbiamo già presentato la tabella dell'operazione $*$ in D_5 (tabella 3.1), ora dobbiamo proporre una τ che soddisfi le proprietà volute. L'applicazione che fornisce questo servizio è

$$\begin{aligned} \tau : D_5 &\longrightarrow D_5 \\ (e, x) &\longmapsto (e, e(1 - x) - 1) \end{aligned} \tag{3.5}$$

e corrisponde alla permutazione σ delle cifre decimali che indichiamo con $(14)(23)(58697)$.

Proposizione 3.2.6. *L'applicazione (3.5) su $D = \{0, 1, \dots, 9\}$ soddisfa il lemma 3.2.1.*

Dimostrazione. Lo verifichiamo per ogni coppia $u, v \in D$ e, per rendere le cose più evidenti, scriviamo la tabella per $\tau(u) * v$ in forma matriciale (semplicemente si

applicano le permutazioni di τ alle righe della tabella di $*$). Ora questa nuova tabella ha valore $\tau(x) * y$ nella casella di posizione (x, y). Guardando la tabella risulta che le entrate in posizione (x, y) sono sempre differenti da quelle in posizione (y, x) a meno che $x = y$, quindi $\tau(x) * y \neq \tau(y) * x$ a meno che $x = y$. □

Teorema 3.2.7. *Sia τ la permutazione* $(14)(23)(58697)$ *sulle cifre di* $D = \{0, 1, \ldots, 9\}$ *e sia $*$ l'operazione del gruppo diedrale su D. Il calcolo del check digit p per un numero di cifre $d_k, d_{k-1}, \ldots, d_1$ come $p = [\tau^k(d_k) * \cdots * \tau(d_1)]^{-1}$ determina una cifra di controllo in grado di individuare tutti gli errori di single digit e tutti gli errori derivanti dalla trasposizione di cifre adiacenti.*

Esempio 3.2. Calcoliamo il check digit per il numero 1793:

$$p = \left[\sigma^4(1) * \sigma^3(7) * \sigma^2(9) * \sigma(3)\right]^{-1} = [1 * 6 * 5 * 2]^{-1} = [4]^{-1} = 1.$$

Il numero protetto sarà quindi 17931. Capire se un numero è corretto con questo metodo è agevole. Il numero 17931 è corretto perché $\sigma^4(1)*\sigma^3(7)*\sigma^2(9)*\sigma(3)*1$ è pari a zero. Invece l'errore in 19731 sarebbe rintracciato perché $\sigma^4(1)*\sigma^3(9)*\sigma^2(7)*\sigma(3)*1$ è pari a $4 \neq 0$.

I contenuti di questo paragrafo sono tratti da [3] e [4].

3.3 Esempi di schemi e algoritmi

I criteri con cui gli schemi vengono solitamente analizzati e valutati sono vari. Per esempio, si può soppesare il fatto che la sequenza sia interamente composta di caratteri numerici, che il metodo funzioni per diverse lunghezze di codice o che non sia ambiguo. Il requisito fondamentale resta però la capacità di rintracciare più errori possibili, focalizzandosi prima di tutto su quelli più frequenti, ossia quelli di cifra singola e di trasposizione.

- **Numero dei vaglia postali in USA**
  Il numero di identificazione è interamente composto di numeri, ha una lunghezza fissata di 11 caratteri e il check digit è l'ultima cifra. Se $n = d_1 d_2 \ldots d_{10} p$,

le sequenze accettate soddisfano la formula

$$\sum_{i=1}^{10} d_i \equiv p \pmod 9$$

Questo metodo individua quasi tutti gli errori di cifra singola ma davvero pochi errori di trasposizione [8].

Esempio 3.3. $n = 67021200988$ è corretto poiché $6+7+0+2+1+2+0+0+9+8 = 35 \equiv 8 \pmod 9$. Dopodiché anche il numero $n = 76021200988$ è accettato quindi questo tipo di errore non è sempre individuato.

- **Codice universale del prodotto (UPC)**
  Assieme al codice a barre EAN, l'UPC è il codice a barre utilizzato principalmente per la scansione di unità commerciali al punto di vendita.
  È interamente composto di numeri e ha una lunghezza fissa di 12 caratteri. Se scriviamo $n = d_1 d_2 \dots d_{11} p$, allora la prima cifra indica il particolare sistema numerico da utilizzare, il codice del produttore è contenuto tra d_2 e d_6, i numeri da d_7 a d_{11} rappresentano il codice del prodotto mentre il check digit è in dodicesima posizione. I numeri accettati soddisfano la formula

  $$3\cdot d_1 + d_2 + 3\cdot d_3 + d_4 + 3\cdot d_5 + d_6 + 3\cdot d_7 + d_8 + 3\cdot d_9 + d_{10} + 3\cdot d_{11} + p \equiv 0 \pmod{10}$$

  Come accennato nel Capitolo 1, questo metodo individua tutti gli errori di cifra singola e quasi tutti gli errori di trasposizione [2] e [8].
  Se $\dots a \dots \to \dots b \dots$ paragoniamo i calcoli:

  $$\begin{aligned} c + 3\cdot a \equiv 0 \pmod{10} \;\&\; c + 3\cdot b &\equiv 0 \pmod{10} \\ (c + 3\cdot a) - (c + 3\cdot b) &\equiv 0 \pmod{10} \\ 3\cdot a - 3\cdot b &\equiv 0 \pmod{10} \\ 3\cdot (a - b) &\equiv 0 \pmod{10} \\ a - b &\equiv 0 \pmod{10} \\ a &\equiv b \pmod{10}. \end{aligned}$$

  Se $\dots ab \dots \to \dots ba \dots$ paragoniamo i calcoli:

$$c + 3 \cdot a + b \equiv 0 \pmod{10} \;\&\; c + 3 \cdot b + a \equiv 0 \pmod{10}$$
$$(c + 3 \cdot a + b) - (c + 3 \cdot b + a) \equiv 0 \pmod{10}$$
$$3 \cdot a + b - 3 \cdot b - a \equiv 0 \pmod{10}$$
$$2 \cdot a - 2 \cdot b \equiv 0 \pmod{10}$$
$$2 \cdot (a - b) \equiv 0 \pmod{10}$$
Non individuato per $|a - b| = 5$.

Esempio 3.4. $n = 053600100540$ è corretto perché $3 \cdot 0 + 5 + 3 \cdot 3 + 6 + 3 \cdot 0 + 0 + 3 \cdot 1 + 0 + 3 \cdot 0 + 5 + 3 \cdot 4 + 0 \equiv 0 \pmod{10}$. Il numero $n = 153600100540$ è sbagliato perché $3 \cdot 1 + 5 + 3 \cdot 3 + 6 + 3 \cdot 0 + 0 + 3 \cdot 1 + 0 + 3 \cdot 0 + 5 + 3 \cdot 4 + 0 = 43 \equiv 3 \pmod{10}$. Mentre $n = 503600100540$ è accettato ma in realtà è sbagliato infatti $3 \cdot 5 + 0 + 3 \cdot 3 + 6 + 3 \cdot 0 + 0 + 3 \cdot 1 + 0 + 3 \cdot 0 + 5 + 3 \cdot 4 + 0 = 43 \equiv 3 \pmod{10}$.

- **Numero di riferimento internazionale del libro (ISBN-10)**
  La sua forma generale è $n = d_1 d_2 \dots d_9 p$, ha una lunghezza fissata di 10 caratteri, è quasi completamente composto da cifre numeriche e la posizione del check digit è 10. I numeri corretti verificano la formula

$$\sum_{i=1}^{10} (11 - i) \cdot d_i \equiv 0 \pmod{11}.$$

  In casi particolari compare anche la lettera X (corrispondente al 10 nei numeri romani). Essa può stare solo nell'ultima posizione e rappresenta il resto quando, essendo in aritmetica modulo 11, è pari a 10 (è possibile trovare anche una A al posto della X).
  Il metodo individua entrambi i tipi di errore che stiamo valutando e, in più, il 100% dei jump twin errors ma non rintraccia il 100% degli errori fonetici e nemmeno il 100% dei twin errors [2] e [8].

  Esempio 3.5. $n = 0673356949$ è corretto poiché $10 \cdot 0 + 9 \cdot 6 + 8 \cdot 7 + 7 \cdot 3 + 6 \cdot 3 + 5 \cdot 5 + 4 \cdot 6 + 3 \cdot 9 + 2 \cdot 4 + 1 \cdot 9 = 242 \equiv 0 \pmod{11}$.
  $m = 068486875\text{X}$ è corretto poiché $10 \cdot 0 + 9 \cdot 6 + 8 \cdot 8 + 7 \cdot 4 + 6 \cdot 8 + 5 \cdot 6 + 4 \cdot 8 + 3 \cdot 7 + 2 \cdot 5 + 1 \cdot 10 = 297 \equiv 0 \pmod{11}$.

- **Schema dell'IBM**

  Questo numero di identificazione è composto di soli numeri, ha una lunghezza n variabile e la cifra di controllo sta in ultima posizione. Se σ è la permutazione data dal ciclo $(0)(124875)(36)(9)$, per ogni stringa di dati $d_1 d_2 \dots d_{n-1}$ assegniamo il check digit p in modo che

  $$\sigma(d_1) + d_2 + \sigma(d_3) + d_4 + \cdots + \sigma(d_{n-1}) + d_n \equiv 0 \pmod{10} \quad \text{se } n \text{ è pari}$$

  oppure

  $$d_1 + \sigma(d_2) + d_3 + \sigma(d_4) + \cdots + \sigma(d_{n-1}) + d_n \equiv 0 \pmod{10} \quad \text{se } n \text{ è dispari}$$

  Di nuovo come accennato nel Capitolo 1, questo metodo individua tutti gli errori di cifra singola e quasi tutti gli errori di trasposizione [2] e [8].

  Esempio 3.6. $n = 76592146$ allora p soddisfa $\sigma(7) + 6 + \sigma(5) + 9 + \sigma(2) + 1 + \sigma(4) + 6 \equiv 0 \pmod{10}$ ovvero $5 + 6 + 1 + 9 + 4 + 1 + 8 + 6 \equiv 0 \pmod{10}$.

- **Schema di Verhoeff**

  Anche questo metodo sfrutta solo caratteri numerici, ha una lunghezza variabile n e porta il check digit in ultima posizione. Ciò che lo caratterizza è l'utilizzo del gruppo non abeliano di ordine 10. Il numero già dotato di check digit $d_1 d_2 \dots d_n p$ viene accettato quando soddisfa la formula

  $$\tau^{n-1}(d_1) * \tau^{n-2}(d_2) * \cdots * \tau^2(d_{n-2}) * \tau(d_{n-1}) * p = 0 \quad \text{in } D_5,$$

  dove τ è la permutazione $(0)(14)(23)(56789)$ e τ^m è definita ricorsivamente come $\tau^0(a) = a$ e $\tau^m(a) = \tau^{m-1}(\tau(a))$ per $m > 1$, mentre la $*$ dello schema è l'operazione nel gruppo diedrale di ordine 10 (tabella 3.1) per cui il risultato della "somma" di due cifre sarà sempre una cifra singola.

  Dato che τ è una permutazione vale $\tau^i(a) \neq \tau^i(b)$ se $a \neq b$, di conseguenza questo schema è in grado di individuare tutti gli errori di singola cifra; inoltre poiché $\tau(a) * b \neq \tau(b) * a$ se $a \neq b$ segue che anche tutti gli errori di trasposizione che coinvolgono cifre adiacenti vengono rintracciati. In realtà esso riconosce tutte le tipologie di errori discusse finora [2], [8] e [10].

Esempio 3.7. Consideriamo un numero di lunghezza $n = 5$, per esempio 46824, e vediamo se verifica il metodo. Applichiamo la formula $\tau^{5-1}(4) + \tau^{5-2}(6) + \tau^{5-3}(8) + \tau^{5-4}(2) + \tau^{5-5}(4) = 4 + 9 + 5 + 3 + 4 = 8 + 5 + 3 + 4 = 3 + 3 + 4 = 1 + 4$ che è pari a 0 in D_5 quindi è corretto.

Conclusioni

In questa tesi abbiamo mostrato come la conoscenza di strutture algebriche semplici come i gruppi ciclici e i gruppi diedrali ponga fondamenta solide per metodi di check digit efficaci.

Abbiamo studiato le tipologie di errori e la frequenza con cui occorrono; abbiamo illustrato come impostare i codici in base a ciò che vogliamo individuare e alla probabilità di successo della ricerca. Successivamente, abbiamo visto in che modo sfruttare le proprietà offerte dall'algebra per costruire sistemi sempre più efficaci e rispondenti alle esigenze di affidabilità e integrità, mostrando, infine, un metodo sempre in grado di rintracciare tutti i possibili errori.

Bibliografia

[1] M. Artin, *Algebra*, New Jersey by Prentice-Hall, Inc. Simon & Schuster Company Englewood Cliffs, 1991, cap. 5

[2] J. A. Gallian, *The Mathematics of Identification Numbers*, The College Mathematics Journal, Vol. 22, No. 3 (May, 1991), pp. 194-202, Mathematical Association of America

[3] H. P. Gumm, *Encoding of Numbers to Detect Typing Errors*, Int. J. Appl. Engng Ed. Vol. 2, No. 1, pp.61-65, 1986, Pergamon Journals Ltd

[4] H. P. Gumm, *A New Class of Check Digit Methods for Arbitrary Number Systems*, IEEE Transactions on Information Theory, Vol. IT-31, No. 1, January 1985

[5] R. W. Hamming, *Error Detecting and Error Correcting Codes*, The Bell System Technical Journal, Vol. XXIX, April 1950, No. 2

[6] Indian Institute of Technology, Kharagpur, NPTEL online, *Software Engineering, Computer Science and Engineering*, Module 3, Lesson 2

[7] T. W. Judson, S. F. Austin, *Abstract Algebra, Theory and Applications*, State University, August 2010

[8] J. Kirtland, *Identification Numbers and Check Digit Schemes*, The Mathematical Association of America Publications, March 2001

[9] J. Verhoeff, *Error Detecting Decimal Codes*, Mathematical Centre Tracts, Amsterdam, 1969

[10] S. J. Winters, *Error Detecting Schemes Using the Dihedral Group*, UMAP Journal, 11(4), 1990

Elenco delle figure

Elenco delle tabelle